Multiplicative Differentials

Nikita Borisov, Monica Chew, Rob Johnson, and David Wagner

University of California at Berkeley

Abstract. We present a new type of differential that is particularly suited to analyzing ciphers that use modular multiplication as a primitive operation. These differentials are partially inspired by the differential used to break Nimbus, and we generalize that result. We use these differentials to break the MultiSwap cipher that is part of the Microsoft Digital Rights Management subsystem, to derive a complementation property in the xmx cipher using the recommended modulus, and to mount a weak key attack on the xmx cipher for many other moduli. We also present weak key attacks on several variants of IDEA. We conclude that cipher designers may have placed too much faith in multiplication as a mixing operator, and that it should be combined with at least two other incompatible group operations. □

1 Introduction

Modular multiplication is a popular primitive for ciphers targeted at software because many CPUs have built-in multiply instructions. In memory-constrained environments, multiplication is an attractive alternative to S-boxes, which are often implemented using large tables. Multiplication has also been quite successful at foiling traditional differential cryptanalysis, which considers pairs of messages of the form $(x, x \oplus \Delta)$ or $(x, x + \Delta)$. These differentials behave well in ciphers that use xors, additions, or bit permutations, but they fall apart in the face of modular multiplication. Thus, we consider differential pairs of the form $(x, \alpha x)$, which clearly commute with multiplication. The task of the cryptanalyst applying multiplicative differentials is to £nd values for α that allow the differential to pass through the other operations in a cipher.

It is well-known that differential cryptanalysis can be applied with respect to any Abelian group, with the group operation de£ning the notion of difference between texts. However, researchers have mostly ignored multiplicative differentials, i.e., differentials over the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^*$, perhaps because it was not clear how to combine them with basic operations like xor. In this paper, we develop new techniques that make multiplicative differentials a more serious threat than previously recognized.

A key observation is that in certain cases, multiplicative differentials can be used to approximate bitwise operations, like xor, with high probability. As we will see in Section 4, for many choices of n there exists a Δ^n such that $-1 \cdot x \bmod n = x \oplus \Delta^n$ with non-negligible probability. Similarly, $2x \bmod 2^n$ is simply a left-shift operation. It is therefore possible to analyze how these differentials interact with other operations that are normally thought incompatible with multiplication, such as xor and bitwise permutations.

Cipher	Complexity [Data]	[Time]	[Keys]	Comments
Nimbus	2^8 CP	2^{10}	all	see [4] (previously known)
xmx (standard version)	2 CP	2	all	mult. complementation property (new)
xmx (challenge version)	2^{33} CP	2^{33}	2^{-8}	multiplicative differentials (new)
MultiSwap	2^{13} CP	2^{25}	all	multiplicative differentials (new)
MultiSwap	2^{22} KP	2^{27}	all	multiplicative differentials (new)
IDEA-X	2^{38} CP	2^{36}	2^{-16}	multiplicative differentials (new)

Table 1. A summary of some cryptanalytic results using multiplicative differentials. The attacks on xmx are distinguishing attacks with advantages close to one; the remaining attacks are key-recovery attacks. All attacks are on the full ciphers; we do not need to consider reduced-round variants. "CP" denotes chosen plaintexts, and "KP" denotes known plaintexts.

After reviewing previous work in Section 2, we give two examples using the ciphers xmx [11] and Nimbus [8] to convey the ¤avor of these attacks in Section 3. In Section 4, we generalize these ideas and catalogue several common cipher primitives that preserve multiplicative differentials. We then focus on speci£c ciphers. Section 5 presents many moduli, including the xmx challenge modulus, that admit large numbers of weak keys in xmx. In Section 6, we examine the MultiSwap cipher [12], which is used in Microsoft's Digital Rights Management system, and show that it is extremely vulnerable to multiplicative differential cryptanalysis. In Section 7, we study several IDEA [7] variants obtained by replacing additions with xors and show that these variants are vulnerable to weak key attacks using multiplicative differentials. As an example, we show that IDEA-X, a version of IDEA derived by replacing all the additions with xors, is insecure. This suggests that multiplicative differentials may yield new attacks on IDEA. Table 1 summarizes the attacks developed in this paper.

2 Related Work

In this paper, we analyze the xmx cipher, originally proposed by M'Raihi, Naccache, Stern and Vaudenay [11]. We also look at Nimbus, which was proposed by Machado [8] and broken by Furman [4]. IDEA was £rst proposed by Lai, Massey and Murphy [7]. Meier observed that part of the IDEA cipher often reduces to an af£ne transformation, and used this to break 2 rounds using differential cryptanalysis [10]. Daemen, Govaerts, and Vandewalle observed that $-x \bmod 2^{16}+1 = x \oplus 11\cdots101$ whenever x_1, the second least signicant bit of x, is 1[2]. They showed that if certain IDEA subkeys are ± 1, the algorithm can be broken with differential cryptanalysis. We use the same observation to £nd weak keys for a variant of IDEA in Section 7. The class of weak keys we £nd is much larger (2^{112} keys versus 2^{51} keys), but they are otherwise unrelated. The newest cipher we look at, MultiSwap, was designed by Microsoft and subsequently reverse-engineered and published on the Internet under the pseudonym Beale Screamer [12].

Differential cryptanalysis was invented by Biham and Shamir [1]. In the present paper, we apply the ideas of differential cryptanalysis using a non-standard group operation: multiplication modulo n. Daemen, van Linden, Govaerts, and Vandewalle have performed a very thorough analysis of multiplication mod $2^\ell - 1$, how it relates to elementary bit-operations, and its potential for foiling differential cryptanalysis [3].

In Section 6 we use the multiplicative homomorphism $(\mathbb{Z}/2^{32}\mathbb{Z})^* \to (\mathbb{Z}/2^{16}\mathbb{Z})^*$ to recover MultiSwap keys ef£ciently. This technique is the multiplicative equivalent of Matsui's linear cryptanalysis [9]. In a similar vein, Harpes, Kramer and Massey applied the quadratic residue multiplicative homomorphism QR: $(\mathbb{Z}/n\mathbb{Z})^* \to \mathbb{Z}/2\mathbb{Z}$, for $n = 2^{16} + 1$, to attack IDEA [5]. Kelsey, Schneier and Wagner used the reduction map $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ (a ring homomorphism), for $n = 2^\ell - 1$ and m dividing n, in cryptanalysis[6].

3 Two Examples

To illustrate some of the ideas behind our attacks, we give two examples of using multiplicative differentials to cryptanalyze simple ciphers. Throughout the paper, x_i will represent the ith bit of x, and x_0 will denote the least signi£cant bit of x.[1]

Cryptanalysis of xmx. As a £rst example, we demonstrate a complementation property for the "standard" version of the xmx cipher [11], which operates on ℓ-bit blocks using two basic operations: multiplication modulo n and xor. The ith round of the cipher is

$$f(x, k_{2i-1}, k_{2i}) = (x \circ k_{2i-1}) \times k_{2i} \bmod n,$$

where the binary operator "$\circ$" is de£ned by

$$x \circ k_{2i-1} = \begin{cases} x \oplus k_{2i-1} & \text{if } x \oplus k_{2i-1} < n \\ x & \text{otherwise.} \end{cases}$$

The cipher has an output termination phase that may be viewed as an extra half-round, so the entire algorithm is

$$\mathsf{xmx}(x) = (f(f(\cdots f(x, k_1, k_2)\cdots), k_{2r-3}, k_{2r-2}), k_{2r-1}, k_{2r}) \circ k_{2r+1}.$$

where r counts the number of rounds.

In the paper introducing xmx [11], the designers recommend selecting $n = 2^\ell - 1$.[2] The curious thing about this choice of n is that for all x,

$$x \oplus n = -x \bmod n.$$

This is a consequence of the following simple observation: if $0 \le x, y < 2^\ell - 1$, then $x+y = 2^\ell - 1$ if and only if $x \oplus y = 2^\ell - 1$. As a result, this differential will be preserved with probability 1 through the entire cipher, giving a complementation property

$$\mathsf{xmx}(-x \bmod n) = -\mathsf{xmx}(x) \bmod n.$$

[1] However, for convenience, we will use k_i to denote a cipher's ith subkey, not the ith bit of k.

[2] Actually, at one point the authors suggest that n be secret, but later state, "Standard implementations should use ... $\ell = 512$, $n = 2^{512} - 1$." For this reason, we call this the "standard" version of xmx, as opposed to the "challenge" version discussed later in this paper.

After describing the basic cipher, the xmx designers suggest several possible extensions, including rotations and other bit permutations. None of these enhancements would destroy this complementation property.

We analyze other versions of xmx later; see Section 5.

Cryptanalysis of Nimbus. As a second example, we explain how the framework of multiplicative differentials can be used to better understand a previously known attack on Nimbus. Nimbus accepts 64-bit blocks, and its ith round is

$$f(x) = k_{2i+1} \times \textit{rev}(x \oplus k_{2i}) \bmod 2^{64},$$

where *rev*() reverses the bits in a 64-bit word. The subkeys k_{2i+1} must be odd for the cipher to be invertible.

At FSE2001, Furman used the xor differential $011 \cdots 10 \longrightarrow 011 \cdots 10$, which passes through one round of Nimbus whenever $t = \textit{rev}(x \oplus k_{2i})$ is odd, to launch a devastating attack on this cipher [4].

Furman's xor differential may appear mysterious at £rst, but can be readily explained using the language of multiplicative differentials. Whenever t is odd,

$$t \oplus 11 \cdots 10 = -t \bmod 2^{\ell}.$$

(This is a standard fact from two's complement arithmetic, and follows from the earlier observation that $(t \oplus 11 \cdots 11) + t = 2^{\ell} - 1$.) So Furman's differential pairs $(x, x \oplus 011 \cdots 10)$ are in fact pairs (x, x^*) where $x^* = -x \bmod 2^{63}$ but $x^* \neq -x \bmod 2^{64}$, a property that obviously survives multiplication by k_{2i+1} whenever k_{2i+1} is odd. In other words, Furman's xor differential is equivalent to the multiplicative differential

$$-1 \longrightarrow -1 \qquad \text{(with probability } 1/2\text{)},$$

taken $\bmod\ 2^{63}$, with explicit analysis of the high bit to ease propagation through the *rev*() operation.

Discussion. The complementation property of standard xmx has not been previously described, despite xmx's relative maturity. The attack on Nimbus was previously described using xor differentials, but is neatly summarized in our new framework for multiplicative differentials. We believe these two examples motivate further study of multiplicative differentials, and the remainder of this paper is dedicated to this task.

4 New Differentials

Most of the conclusions in this section are summarized in Table 2.

The xmx example in Section 3 used the multiplicative difference $\alpha = -1$, because $-x \bmod 2^{\ell} - 1 = x \oplus 11 \cdots 1$. Thus the multiplicative differential pair $(x, -x)$ is equivalent to the xor differential pair $(x, x \oplus 11 \cdots 1)$. In the Nimbus example, the modulus is of the form 2^{ℓ} instead of $2^{\ell} - 1$, so the identity between the multiplicative and xor differentials does not hold. However, there is an approximate identity $-x \bmod 2^{\ell} = x \oplus 11 \cdots 10$, which holds whenever x is odd, or equivalently, when $x_0 = 1$.

Operation	Modulus	multiply by α	xor	rotate	bit perm σ
multiply by -1	$2^\ell - 1$	1	1	1	1
multiply by -1	2^ℓ	1	$\frac{1}{2}$	0	$z(\sigma)$
multiply by -1	n	1	$2^{-c(n)}$	–	–
multiply by 2	$2^\ell - 1$	1	0	1	–
multiply by 2	2^ℓ	1	0	$\frac{1}{4}$	$2^{-\omega(\sigma)-z(\sigma)}$
reduction mod 2^k	2^ℓ	1	1	–	–

Table 2. A partial list of the operations we consider. Each entry in the table speci£es the probability that the two operations commute. See Proposition 1 for an explanation of $c(n)$. See Proposition 3 for the de£nitions of $z(\sigma)$ and $\omega(\sigma)$. An entry of "0" indicates the probability is negligible, and a "–" means we do not investigate this combination.

n	1	1	1	1	0	0	0	0	0	1	1	1	0	0	1
x	x_{14}	x_{13}	x_{12}	0	x_{10}	x_9	x_8	x_7	1	x_5	x_4	0	x_2	1	x_0
Δ^n	1	1	1	0	1	1	1	1	0	1	1	0	1	0	1
$x + x \oplus \Delta^n$	1	1	1	1	0	0	0	0	0	1	1	1	0	0	1

Fig. 1. The modulus $n = 30777$, the bit-constraints on values of x for which $x + (x \oplus \Delta^n) = n$, and Δ^n. See Proposition 1 for a precise de£nition of Δ^n.

To generalize the multiplicative/xor correspondence exploited in these two examples, £rst observe that every ℓ-bit modulus, n, can be divided into strings of the form $11\cdots1$ and strings of the form $100\cdots0$. As an example, the 15-bit modulus $n = 30777$ is divided into such substrings in Figure 1.

For each segment of the modulus of the form $11\cdots1$, we use the xor differential $11\cdots1$. For the segments of the modulus of the form $100\cdots0$, we use the xor differential $011\cdots10$. Suppose $n_k\cdots n_j$ is one of the segments of n of the form $100\cdots0$. Then we also require that $x_j = 1$ and $x_k = 0$. The constraint that $x_j = 1$ serves the same purpose as the constraint that x be odd in the Nimbus differential: it ensures that when x and $x \oplus \Delta^n$ are added together, a chain of carries is started at bit j. The requirement that $x_k = 0$ assures that no carry bits propagate past bit k when x and $x \oplus \Delta^n$ are added together. In the example, bit i of x is constrained if and only if bit i of Δ^n is 0. This is always true because of the symmetry between x and $-x$.

The above scheme works by controlling the carry bits when x and $x \oplus \Delta^n$ are added together. It ensures that, for each substring of the modulus of the form $10\cdots0$, a carry chain is started at the low bit and terminated at the high bit. Starting and stopping carry chains necessitates imposing constraints on x, and if two substrings of the form $10\cdots0$ are adjacent, it is more ef£cient to simply ensure that the carry chain from the £rst substring propagates to the second. Analogously, if the modulus contains a substring of the form $11\cdots1011\cdots1$, then the above method will start a carry chain, only to terminate it at the next bit. A more ef£cient approach would ensure that no carry ever started. Algorithm 1, which computes an optimal value of Δ^n for a given n, incorporates these improvements. The algorithm also outputs ω^n and ν^n, which represent the bits of x constrained to 0 and 1, respectively.

Algorithm 1 *Compute the optimal $\Delta = \neg(\omega \vee \nu)$.*

```
best-differential(n)
  c ← 0, ω,ν ← 00···0
  for i = 0,...,length(n)-2
    switch (n_{i+1}, n_i, c)
      case (0,0,0) // Begin a carry chain by requiring x_i = 1.
            ν_i ← 1, c ← 1
      case (0,1,1) // Force carry propagation by requiring x_i = 1.
            ν_i ← 1
      case (1,0,0) // Force no carry by requiring x_i = 0.
            ω_i ← 1
      case (1,1,1) // End carry chain by requiring x_i = 0.
            ω_i ← 1, c ← 0
      default // No change to carry bit. No constraint on x.
  if c = 1 then ω_{ℓ-1} ← 1
  Δ = ¬(ω ∨ ν)
  output (Δ,ω,ν)
```

To determine the probability that a randomly selected $x \in \mathbb{Z}/n\mathbb{Z}$ satis£es the bit-constraints described above, let $c(n)$ be the number of 0 bits in Δ^n (i.e., the number of bits of x that are constrained). Then x will satisfy these constraints with probability at least $2^{-c(n)}$. To see why this is only a lower bound, consider the modulus $n = 1001$ (base 2). The constraints derived from this modulus are $x_3 = 0$ and $x_1 = 1$. However, only one value of $x \in \mathbb{Z}/n\mathbb{Z}$ fails to satisfy $x_3 = 0$, so this constraint is nearly vacuous. The following proposition formalizes this discussion:

Proposition 1. *Let n be an ℓ-bit modulus. Let the ℓ-bit words ω^n, ν^n be the result of Algorithm 1, and let $\Delta^n = \neg(\omega^n \vee \nu^n)$. Take any $x \in \mathbb{Z}/n\mathbb{Z}$. De£ne:*

$$C_n(x) = \begin{cases} -1 & \textit{if } x \wedge \omega^n = 0 \textit{ and } \neg x \wedge \nu^n = 0 \\ 1 & \textit{otherwise.} \end{cases}$$

Then $C_n(x) = -1$ if and only if $-x \bmod n = x \oplus \Delta^n$. By symmetry, $C_n(-x) = C_n(x)$. Further, de£ne $c(n)$ to be the number of 0 bits in Δ^n. Then, for a uniformly distributed $x \in \mathbb{Z}/n\mathbb{Z}$, $C_n(x) = -1$ with probability at least $2^{-c(n)}$. Finally, for any Δ', $\Pr[x \oplus (-x \bmod n) = \Delta'] \leq \Pr[x \oplus (-x \bmod n) = \Delta^n]$.

The Nimbus attack uses the slight tweak of considering pairs (x, x^*) such that $x^* = -x \bmod 2^{\ell-1}$ but not mod 2^ℓ. Generalizing this gives a truncated multiplicative differential.

Proposition 2. *Suppose*

$$x^* = x \oplus (a_{\ell-1}a_{\ell-2}\cdots a_m 11 \cdots 10),$$

where each a_i stands for any single bit, and suppose moreover that x is odd. If k is odd, then

$$k \times x^* = (k \times x) \oplus b_{\ell-1}b_{\ell-2}\cdots b_m 11 \cdots 10,$$

where the multiplication is modulo 2^ℓ. Additionally, $a_m = b_m$.

Until now, we have only discussed multiplicative differential pairs $(x, -x)$, but the cryptanalysis of MultiSwap uses pairs of the form $(x, 2x \bmod 2^{32})$. One of the basic operations in MultiSwap is to swap the two 16-bit halves of a 32-bit word. The multiplicative relation $(x, 2x)$ is preserved through this operation whenever $x_{15} = x_{31} = 0$.

An arbitrary bit permutation σ can cause two types of problems for the multiplicative differential 2. First, it can disturb the consecutive ordering of the bits. Because multiplication by 2 is just a left-shift, it's not surprising that the bit ordering comes into play. Second, σ may place some bit i in position 0. If σ is to commute with multiplication by 2, then the value of bit i must be 0. These notions are summarized in the following proposition:

Proposition 3. *Let σ be a permutation of the set $\{0, \ldots, \ell-1\}$, and let $\hat{\sigma}$ be the induced function on ℓ-bit words given by $\hat{\sigma}(x) = x_{\sigma(\ell-1)} x_{\sigma(\ell-2)} \cdots x_{\sigma(0)}$. Then*

$$\Pr\left[\hat{\sigma}(2x) = 2\hat{\sigma}(x)\right] = 2^{-\omega(\sigma) - z(\sigma)}$$

where

$$\omega(\sigma) = \#\{j \in 0, \ldots, \ell - 2 | \sigma(j + 1) \neq \sigma(j) + 1\}$$

and

$$z(\sigma) = \begin{cases} 0 & \textit{if } \sigma(0) = 0 \\ 1 & \textit{otherwise.} \end{cases}$$

Intuitively, $\omega(\sigma)$ counts the number of times that σ disturbs the consecutive ordering of the bits, and $z(\sigma)$ tests whether σ places bit $i \neq 0$ in position 0. So, for example, the $(x, 2x)$ differential survives rotations with probability $\frac{1}{4}$ independent of the amount of rotation. Also, dividing a word into k chunks, such as dividing a 32-bit word into 4 bytes, and permuting the chunks leaves the differential undisturbed with probability 2^{-k}.

Multiplicative differentials are compatible with many other operations. Reversing the bits in a word transforms the pair $(x, 2x \bmod 2^\ell)$ into $(x, x/2 \bmod 2^\ell)$ with probability 1. Multiplicative differentials may even survive addition in some cases, since $\alpha \times a + \alpha \times b = \alpha \times (a + b)$. Finally one may consider differentials in which part of the differential is de£ned using multiplication, and part is de£ned using some other operation. For example, if a cipher operates on 64-bit blocks (a, b, c, d), where a, b, c, and d are 16-bit subblocks, we may want to consider differential pairs (a, b, c, d) and (a^*, b^*, c^*, d^*) where $a^* = \alpha \times a$, $b^* = b \oplus \Delta$, $c^* = c \oplus \Delta$, and $d^* = \alpha \times d$. In other words, the differences $(\alpha, \Delta, \Delta, \alpha)$ are elements of the group $(\mathbb{Z}/2^{16}\mathbb{Z})^* \times (\mathbb{Z}/2\mathbb{Z})^{16} \times (\mathbb{Z}/2\mathbb{Z})^{16} \times (\mathbb{Z}/2^{16}\mathbb{Z})^*$. When there can be no confusion as to the groups in question, we simply refer to these as "hybrid" differentials.

5 xmx

We can now apply our new understanding to £nd differentials for a large class of moduli in the xmx cipher.

We describe the analysis using the parameters given in the "xmx challenge" [11]. This cipher has 8 rounds, 256-bit blocks, and modulus $n = (2^{80} - 1) \cdot 2^{176} + 157$, which is the smallest prime whose 80 most signi£cant bits are all 1. Written in binary, this modulus is

$$n = \overbrace{11\cdots 1}^{80}\,\overbrace{00\cdots 0}^{168}\,10011101,$$

which has $c(n) = 4$. From Proposition 1, whenever x is of the form

$$x = x_{255}x_{254}\cdots x_{177}0x_{175}\cdots x_8 1x_6 1x_4x_3x_2 0x_0$$

(i.e. whenever $x_{176} = 0, x_7 = 1, x_5 = 1$ and $x_1 = 0$) then $C_n(x) = -1$ and therefore $x \oplus \Delta^n = -x \mod n$, where

$$\Delta^n = \overbrace{11\cdots 1}^{79}\,0\,\overbrace{11\cdots 1}^{167}\,101011101.$$

Recall that Δ^n has a 0 bit in exactly those positions that are constrained in x. If $k \wedge \neg\Delta^n = 0$, then k has a 0 in each constrained bit position, and hence $C_n(x \oplus k) = C_n(x)$.

The key schedule for the xmx challenge cipher is

$$s, s, \ldots, s, s, s \oplus s^{-1}, s, s^{-1}, \ldots, s, s^{-1}$$

where s is a 256-bit number. Suppose $s \wedge \neg\Delta^n = 0$ and $s^{-1} \wedge \neg\Delta^n = 0$. The £rst equation is satis£ed whenever bits 1,5,7 and 176 of s are 0, and hence will be satis£ed with probability 2^{-4}. The second equation establishes similar requirements on the bits of s^{-1}, and will be satis£ed with probability 2^{-4}. So about 2^{-8} of the keys s satisfy these constraints simultaneously. Obviously, if $s \wedge \neg\Delta^n = 0$ and $s^{-1} \wedge \neg\Delta^n = 0$, then $(s \oplus s^{-1}) \wedge \neg\Delta^n = 0$, as well.

Consider one round of xmx using such a weak key, and let a and b be the subkeys for the current round, so that $a = s$ or s^{-1} or $s \oplus s^{-1}$, but it doesn't matter which. Suppose we apply this round of the cipher to the differential pair (x, x^*), where $x \oplus x^* = \Delta^n$ and $C_n(x) = -1$. Then by Proposition 1, $x^* = x \oplus \Delta^n = -x \mod n$. Since $a \wedge \neg\Delta^n = 0$, $C_n(x \oplus a) = -1$, so $x^* \oplus a = x \oplus a \oplus \Delta^n = -(x \oplus a) \mod n$.

We would like to conclude that $-(x \circ a) = x^* \circ a \mod n$, but must consider the two different behaviors of the operator "$\circ$". From the de£nition,

$$x \circ a = \begin{cases} x \oplus a & \text{if } x \oplus a < n \\ x & \text{otherwise.} \end{cases}$$

But by assumption, $x_{176} = a_{176} = 0$. Thus bit 176 of $x \oplus a$ is also 0. Furthermore, bits 255 through 176 of n are all 1. This implies $x \oplus a < n$. Bit 176 of Δ^n is 0, so $x^*_{176} = 0$, and hence $x^* \oplus a < n$ for the same reasons. From this, $-(x \circ a) = -(x \oplus a) = x \oplus a \oplus \Delta^n = x^* \oplus a = x^* \circ a$.

So $x^* \circ a = -(x \circ a) \mod n$. The next step in a round of xmx is multiplication by the second subkey, b, which will preserve this multiplicative relationship. So at the end of one round of xmx, with probability 1, the outputs $y = (x \circ a) \times b \mod n$ and $y^* = (x^* \circ a) \times b \mod n$ will satisfy $y^* = -y \mod n$. However, it's not clear that $C_n(y) = -1$. Since multiplication by b affects each bit of the output in a complicated way, we can assume that y is randomly distributed, and therefore $C_n(y) = -1$ with probability $2^{-c(n)} = 2^{-4}$, by Proposition 1. When $C_n(y) = -1$, $y^* = y \oplus \Delta^n$. Thus an input pair $(x, x \oplus \Delta^n)$ becomes an output pair of the form $(y, y \oplus \Delta^n)$ after one round of encryption with probability $\frac{1}{16}$. This yields the following 1-round iterative xor-differential:

$$\Delta^n \longrightarrow \Delta^n \qquad \text{(with probability } 1/16\text{)},$$

or equivalently, the 1-round iterative multiplicative differential

$$-1 \longrightarrow -1 \qquad \text{(with probability } 1/16\text{)}.$$

The probability of the differential may be much higher for many keys, because there are many $-1 \leftrightarrow \Delta$ correspondences that hold with high probability. For example,

$$x \oplus (-x \bmod n) = \overbrace{11\cdots1}^{78}\,00\,\overbrace{11\cdots1}^{167}\,101011101.$$

whenever $x_1 = 0$, $x_5 = 1$, $x_7 = 1$, $x_{176} = 1$ and $x_{177} = 0$. Thus, if, in addition to the weak key constraints described above, $s_{177} = s^{-1}_{177} = 0$, then the multiplicative differential -1 survives one round of the cipher with probability $2^{-4} + 2^{-5}$. There are many other very similar differentials, and if s satis£es even more weak key constraints, then the -1 differential will survive with even greater probability.

This differential survives 8 rounds of the cipher with probability 2^{-32}. The last half round of the cipher consists of only the "$\circ$" operator, and we've already seen that this differential passes through that operation with probability 1, so the differential survives the whole cipher with probability 2^{-32}. Each right pair, (x, x^*), yields 4 constraints on the bits of $(x \oplus s) \times s \mod n$, the output of the £rst round of the cipher. Although a careful analysis of multiplication mod n may reveal an ef£cient key recovery attack, we leave this as a distinguishing attack.

This analysis easily generalizes to other instances of xmx with different parameters. For any ℓ-bit modulus n that is not a power of 2, we can compute Δ^n and $c(n)$ as described in the previous section. Consider a single round of xmx that uses modulus n, subkeys k_{2i-1} and k_{2i} in the $\circ$ and multiply steps respectively, and suppose $k_{2i-1} \wedge \neg\Delta^n = 0$. Given an input pair $(x, x \oplus \Delta^n)$ where $C_n(x) = -1$, with probability $2^{-c(n)}$ the output of the round for the pair is of the form $(y, y \oplus \Delta^n)$, with $C_n(y) = -1$, by an analysis similar to the one above. Therefore, the differential survives r rounds of the cipher with probability $2^{-c(n)r}$, as long as each subkey used in the "$\circ$" operation satis£es $k_{2i-1} \wedge \neg\Delta^n = 0$. If independent subkeys are used, $2^{-c(n)r}$ of all keys satisfy this weak key condition. If the xmx key schedule is used, $2^{-2c(n)}$ of all keys are weak, since only s and s^{-1} must satisfy the condition.

Whenever the modulus n used in xmx has a highly regular bit pattern—in particular, long sequences of 1's and 0's—$c(n)$ will be small and therefore such a weak key analysis may be of signi£cantly lower complexity than an exhaustive search.

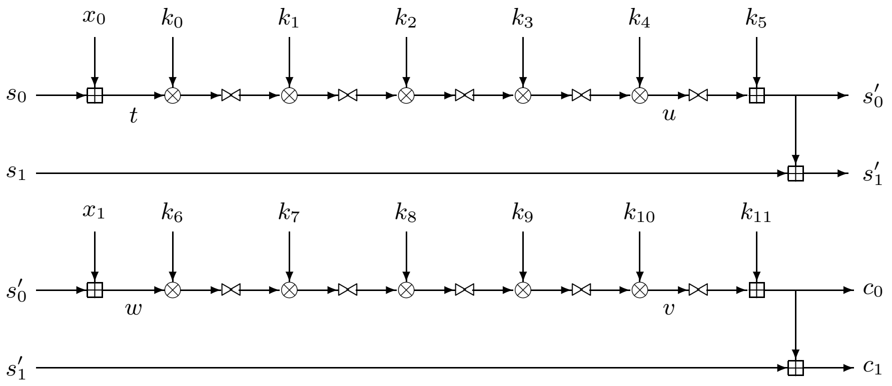


Fig. 2. The MultiSwap cipher. Processing begins with $s_0 = s_1 = 0$ on plaintext x_0, x_1 and proceeds from left to right. The output of the £rst half, s'_0 and s'_1, is fed into the second half to produce ciphertext c_0, c_1. The "$\bowtie$" operator swaps the 16-bit halves of a 32-bit word, "$\boxplus$" represents addition modulo 2^{32}, and "$\otimes$" represents multiplication modulo 2^{32}.

6 MultiSwap

The MultiSwap cipher is used in Microsoft's Digital Rights Management subsystem and was £rst described in a report published on the Internet under the pseudonym Beale Screamer [12]. The cipher, shown in Figure 2, operates entirely on 32-bit words, maintains two words of internal state, s_0 and s_1, and uses 12 32-bit subkeys $k_0, \ldots, k_{11}$. The subkeys $k_0, \ldots, k_4, k_6, \ldots, k_{10}$ must be odd if the cipher is to be invertible. Unless the cipher is being used in some sort of feedback mode, $s_0 = s_1 = 0$; we will assume this in the analysis. This analysis is also applicable when s_0 and s_1 are non-zero but their values are known. No key schedule is described, so we assume the subkeys are all independent. The cipher operates on 64-bit blocks (x_0, x_1) to produce ciphertext (c_0, c_1).

We £rst present a chosen-plaintext attack, and then describe how to convert this to a known-plaintext attack. Consider the algorithm operating on input $(0, x_1)$. Since $s_0 = s_1 = 0$, $t = s_0 + x_0 = 0$. Since $u = 0$ if and only if $t = 0$, u is also 0. Thus $s'_0 = s'_1 = k_5$. After the second half, regardless of the input x_1 the output satis£es $c_1 = c_0 + k_5$. Thus one can derive $k_5 = c_1 - c_0$ with one chosen-plaintext message of the form $(0, x_1)$. Given k_5, one additional message suf£ces to recover k_{11}. With input $(0, -k_5)$, it is still the case that $s'_0 = s'_1 = k_5$. In the second half, though, since $x_1 = -k_5$, $w = s'_0 + x_1 = k_5 + (-k_5) = 0$, which propagates through the multiplications and swaps as before. Thus the output is $c_0 = k_{11}$ and $c_1 = k_5 + k_{11}$. So a 2-message adaptive chosen-plaintext attack exposes k_5 and k_{11}.

Given k_5, we can control the input to the second half of the cipher. To make $w = a$, query the encryption oracle with the plaintext $(0, a - k_5)$. With k_{11}, we can partially decrypt a ciphertext to obtain the intermediate value v. Therefore, we only have to analyze the sequence of multiplications and swaps in the second half of the cipher between w and v. Similarly, we can analyze the sequence between t and u using knowledge of

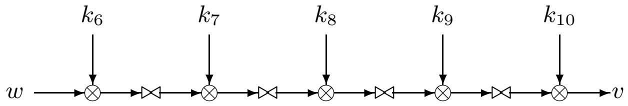


Fig. 3. The second half of the MultiSwap cipher.

k_5 and the fact that $s'_0 = c_1 - c_0 - s_1$. Because this is a chosen-plaintext attack, we have reduced the problem to the system in Figure 3 for which the input, w, can be controlled and the output, v, can be observed. The goal is to recover $k_6, \ldots, k_{10}$.

So we focus only on this fragment of MultiSwap. If this fragment operates on inputs w and $w^* = 2w$, then $k_6 \cdot w^* = k_6 \cdot (2w) = 2(k_6 \cdot w)$. From Proposition 3, $\bowtie(k_6 \cdot w^*) = 2 \cdot \bowtie(k_6 \cdot w)$ whenever bits 15 and 31 of $k_6 \cdot w$ are 0, or $\frac{1}{4}$ of the time. Analyzing the rest of Figure 3 in the same way shows that $v^* = 2v$ with probability $\frac{1}{256}$.

If this condition holds, call $(w, 2w)$ a right pair. Then with high probability bits 15 and 31 of $k_6 \cdot w$ are 0. This is a two-bit condition on $k_6 \cdot w$ that we can use to £lter the set of potential values of k_6; $\frac{1}{4}$ of all k_6 values will pass this test. We can repeat this test for 16 right input pairs $(w_1, 2w_1), \ldots, (w_{16}, 2w_{16})$ chosen uniformly at random, and the probability of a given k_6 value surviving all 16 tests is roughly $(\frac{1}{4})^{16} = 2^{-32}$, so with high probability only one value of k_6 survives.

If $(w, 2w)$ is a right pair, then the multiplicative differential of $w^* = 2w$ must survive each one of the $\bowtie$ operations. Therefore, $k_7 \cdot \bowtie(k_6 \cdot w)$ must have bits 15 and 31 set to 0. Thus the same right pairs can determine k_7, and then k_8 and k_9. At this point we can determine k_{10} from any known-plaintext. Thus 16 right pairs are enough to recover $k_6, \ldots, k_{10}$, and we can obtain the pairs with about 2^{12} chosen plaintexts. Repeating the analysis for $k_0, \ldots, k_4$ breaks the whole cipher with 2^{13} chosen plaintexts. This is surprisingly small considering the large key size.

The work factor of breaking the cipher is quite low. Let $(w_1, 2w_1), \ldots, (w_{16}, 2w_{16})$ be right pairs that determine k_6. By de£nition of being right, bits 15 and 31 of $k_6 \cdot w_i$ are 0 for all i. Observe that bit 15 of $k_6 \cdot w_i$ is independent of bits 16 through 31 of k_6. Thus we can determine the value of the low 16 bits of k_6 independently of the high bits. After discovering the low 16 bits, we can then do the same thing for the upper 16 bits. Since we have to test each half of a key against each right pair, the total number of tests performed is $2 \cdot 2^{16} \cdot 16 = 2^{21}$. Repeating for $k_7, ..., k_9$, and then again for $k_0, \ldots, k_3$ yields a break on the whole cipher requiring $8 \cdot 2^{21} = 2^{24}$ tests. Each test is quite cheap, involving only a multiply, bit-mask, and test for equality.

To convert this to a known-plaintext attack, observe that even without knowledge of k_5 and k_{11}, we can derive the input to the second half of the cipher via $w = c_1 - c_0 + x_1$. Consider a pair of inputs such that the differential $(w, 2w) \longrightarrow (v, 2v)$ holds. Suppose further that $\bowtie(2v) = 2 \cdot \bowtie(v)$. In this case, $c_0 = \bowtie(v) + k_{11}$ and $c_0^* = 2 \cdot \bowtie(v) + k_{11}$; hence, $k_{11} = 2c_0 - c_0^*$. If, on the other hand, $\bowtie(2v) \neq 2 \cdot \bowtie(v)$, there are three possible values for c_0^*: $2c_0 + k_{11} - 1, 2c_0 + k_{11} + 65536, 2c_0 + k_{11} + 65535$. Each of these possibilities suggests an equation for k_{11}; we can try all four equations and see which makes $v^* = 2v$ hold under partial decryption of c_0, c_0^*. Therefore, each right pair suggests the correct value for k_{11}.

So collect 2^{22} known plaintexts which by the birthday paradox will contain 2^{12} pairs whose input to the second half of the cipher is of the form $(w, 2w)$. Each pair is a right pair with probability 2^{-8}, so the correct k_{11} value will be suggested 16 times. Most wrong pairs will suggest a random value for k_{11}, but, because the sequence of multiplies and swaps maintains suf£cient structure, some incorrect values of k_{11} will be suggested with a lower, but still signi£cant probability. In practice, the correct k_{11} will be among the top few, say 8, suggested; since the rest of the analysis is fast, we can repeat it for each of the top 8 suggested values of k_{11} and use trial encryptions to detect the correct one.

With k_{11}, we can now use the same set of pairs to recover $k_6, \ldots, k_{10}$. A similar attack reveals k_5, and then $k_0, \ldots, k_4$. Except for having to repeat the attack for several possible values of k_{11} and k_5, the work factor is about the same as for the previous attacks. Hence the total work for the known plaintext attack is 2^{27}. The storage is also quite small, since we don't have to keep a counter for every possible value of k_{11}, only the ones suggested by a pair. Since the attack uses only about 2^{12} pairs, the storage requirement is about 2^{15} bytes.

To summarize, MultiSwap can be broken with a 2^{13} chosen-plaintext attack requiring 2^{25} work or a 2^{22} known-plaintext attack requiring a work factor of about 2^{27}.

7 IDEA Variants

The IDEA cipher designers deliberately used incompatible group operations to destroy any algebraic relations among the inputs, and this strategy has proven very successful. The basic operations used in IDEA are addition modulo 2^{16}, xor of 16-bit words, and multiplication modulo $2^{16} + 1$.

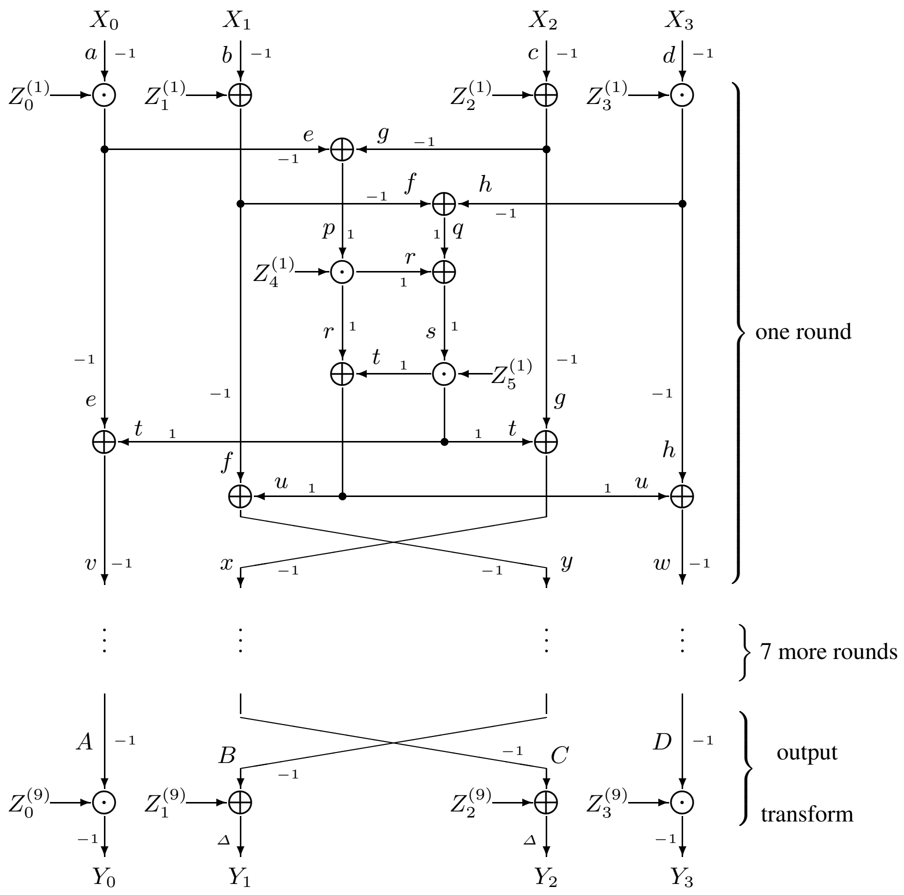
IDEA uses the addition operation in two places: key mixing and the MA-structure. As has been noted before [10], if $a+b < 2^{16}$, then $a+b \mod 2^{16} = a+b \mod 2^{16}+1$, and thus the MA-structure is linear about $\frac{1}{4}$ of the time. We consider a variant of IDEA, which we call IDEA-X, in which all the additions have been replaced by xors. Because of the observations above, it may appear that IDEA-X is an improvement over IDEA, but we show below that IDEA-X has a large class of weak keys for which it is susceptible to multiplicative differential cryptanalysis.

Because of the heavy use of the xor operation in IDEA-X, we use the multiplicative differential $(-1, -1, -1, -1)$. Let $n = 2^{16} + 1$, and $\Delta = 11 \cdots 101$. By Proposition 1, $-x \bmod n = x \oplus \Delta$ if and only if $C_n(x) = -1$. For $n = 2^{16} + 1$, $C_n(x) = -1$ if and only if $x_1 = 1$.[3] The analysis maintains, with non-negligible probability, an invariant on all the intermediate values z and z^* in the cipher. The invariant is that all the intermediate values will satisfy the relation $z^* = (-1)^{z_1} z$. This condition may look mysterious, but it simply means that either $z^* = z$, or $z^* = -z = z \oplus \Delta$. The rest of the analysis is essentially repeated application of the following two rules.

Rule 1 *If $x^* = (-1)^{x_1} \cdot x$, then $kx^* = (-1)^{(kx)_1} \cdot kx$ with probability $\frac{1}{2}$. This is the multiplication rule.*

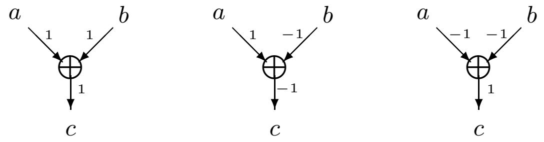
Rule 2 *If $a^* = (-1)^{a_1} \cdot a$ and $b^* = (-1)^{b_1} \cdot b$, then $a^* \oplus b^* = (-1)^{(a \oplus b)_1} \cdot (a \oplus b)$ with probability 1. This is the xor rule.*

[3] Technically, $C_n(x) = -1$ if and only if $x_1 = 1$ and $x_{16} = 0$, but since x is a 16-bit number, the latter condition is vacuous.



X_i : 16-bit plaintext subblock
Y_i : 16-bit ciphertext subblock
$Z_i^{(r)}$: 16-bit key subblock
$\bigoplus$: bit-by-bit exclusive-OR of 16-bit subblocks
$\bigodot$: multiplication modulo $2^{16} + 1$ of 16-bit integers
with the zero subblock corresponding to 2^{16}

Fig. 4. The IDEA-X cipher. All the adds in IDEA have been changed to xors. The diagram is annotated with the path of the multiplicative differential $(-1, -1, -1, -1)$.



(i) (ii) (iii)

Fig. 5. The three cases of the xor rule. The edges are labeled with the multiplicative differentials, e.g. x^*/x. The probability in each case is 1. Recall that differential pairs always satisfy $x^* = (-1)^{x_1} \cdot x$. So in case (iii), $a_1 = b_1 = 1$, and hence $a^* = -a = a \oplus \Delta$ and $b^* = -b = b \oplus \Delta$. Thus $c^* = -a \oplus -b = a \oplus \Delta \oplus b \oplus \Delta = a \oplus b = c$. Furthermore, $a_1 = b_1 = 1$, so $c_1 = 0$. Hence $c^* = (-1)^{c_1} \cdot c$. The other cases are similar.

Both rules are easy to prove. Figure 5 explains the xor rule in more detail.

To demonstrate the use of these rules, consider one round of IDEA-X in which bit 1 of $Z_2^{(1)}$ is 0. This is a weak key condition. We'll look only at the inputs X_0 and X_2. Referring to Figure 4, consider two different executions of the round, one with inputs $X_0 = a$ and $X_2 = c$, the other with inputs $X_0 = -a$ and $X_2 = -c$. Suppose also that $a_1 = c_1 = 1$. By the xor rule, $g_1 = 1$ and $g^* = -g = (-1)^{g_1} g$ with probability 1. By the multiplication rule, $e_1 = 1$ and $e^* = -e = (-1)^{e_1} e$ with probability $\frac{1}{2}$. Combining these two results and applying the xor rule to $e \oplus g$ shows that $p^* = p$ whenever $e_1 = 1$.

If we also assume that bit 1 of $Z_1^{(1)} = 0$ then more of the same sort of reasoning shows that the multiplicative differential

$$(-1, -1, -1, -1) \longrightarrow (-1, -1, -1, -1)$$

survives one round of IDEA-X with probability $\frac{1}{16}$. In order for this differential to work, the input (a, b, c, d) must satisfy $a_1 = b_1 = c_1 = d_1 = 1$. When the differential does successfully pass through the round, the output (v, w, x, y) satis£es $v_1 = w_1 = x_1 = y_1 = 1$. Thus the differential can be iterated.

So we have found an iterative 1-round multiplicative differential that works for keys in which bit 1 of $Z_1^{(i)}$ is 0 and bit 1 of $Z_2^{(i)}$ is 0 in every round. This differential survives 8 rounds of IDEA-X with probability 2^{-32}, and works against 2^{-16} of the keys. The only thing left to consider is the output phase. This phase uses multiplications, which will not disturb the -1 differential, and xors. The differential will survive the xors even without weak key constraints on the subkeys used in the output phase. To see this, consider a differential that has passed 8 rounds; then we have a pair of intermediate texts (A, B, C, D) and (A^*, B^*, C^*, D^*) where $(A^*, B^*, C^*, D^*) = (-A, -B, -C, -D)$. Recall that the -1 multiplicative differential is equivalent to the Δ xor differential. Therefore, $B^* = B \oplus \Delta$ and $C^* = C \oplus \Delta$. This differential survives the £nal xor of the output phase, giving a hybrid differential $(Y_0^*, Y_1^*, Y_2^*, Y_3^*) = (-Y_0, Y_1 \oplus \Delta, Y_2 \oplus \Delta, -Y_3)$ that survives the whole cipher with probability 2^{-32} for 2^{-16} of the keys.

Using this differential to recover keys is relatively straightforward. An attack using 2^{38} chosen plaintexts yields 32 right pairs with high probability. Each right pair (a, b, c, d) and (a^*, b^*, c^*, d^*) establishes the condition that bit 1 of $Z_0^{(1)} \cdot a$ is 1. Just as in the MultiSwap attack, we can use this condition to £lter the possible values of $Z_0^{(1)}$,

Round	Modi£cations
A	The additions in the MA-structure are changed to xors or multiplications
B	The subkey Z_2 is mixed with input X_2 using xor or multiplication
C	The subkey Z_1 is mixed with input X_1 using xor or multiplication The additions in the MA-structure are changed to xors or multiplications

Table 3. A characterization of many IDEA variants which are susceptible to multiplicative differential cryptanalysis.

and given 32 right pairs only two values of $Z_0^{(1)}$ will survive. Unfortunately, whenever $Z_0^{(1)} \cdot a$ satis£es this constraint, so will $-Z_0^{(1)} \cdot a$, so this £lter will leave us with two choices for $Z_0^{(1)}$ which differ by a factor of -1. We can recover $\pm Z_3^{(1)}$ in a similar manner. Each right pair also yields a constraint $r_1 = 0$. Observe that $r = Z_4^{(1)} \cdot (e \oplus Z_2^{(1)} \oplus c)$. After recovering $\pm Z_0^{(1)}$, we can compute $\pm e$. Thus we can compute the correct value of e or $e \oplus \Delta$. Hence we can use this condition to £lter possible values for $(Z_2^{(1)}, Z_4^{(1)})$ and given 32 right pairs only two values will survive. If we guess the wrong value for $Z_0^{(1)}$, we will perform this £ltering with intermediate value $e \oplus \Delta$, and hence will compute $Z_2^{(1)} \oplus \Delta$ instead of $Z_2^{(1)}$. As before, the £lter can only determine $Z_4^{(1)}$ up to a factor of ± 1. We now guess the correct value of $Z_4^{(1)}$ and recover $Z_1^{(1)}$ and $\pm Z_5^{(1)}$ in a manner similar to the recovery of $Z_2^{(1)}$ and $\pm Z_4^{(1)}$. We then guess the correct value of $Z_5^{(1)}$, and recover $\pm Z_0^{(2)}$ the same way we recovered $Z_0^{(1)}$. We £nish by making guesses for $\pm Z_0^{(1)}$, $\pm Z_3^{(1)}$, and $\pm Z_0^{(2)}$ and using trial encryptions to recover $Z_1^{(2)}$ and verify our guesses.

Recovering $Z_1^{(1)}$ and $Z_5^{(1)}$, dominates the analysis time, requiring $2 \cdot 2^{32} \cdot 32 \approx 2^{38}$ trials. Each trial involves one round of IDEA-X, so the work required is equivalent to about 2^{35} IDEA-X encryptions. A generous estimate of the rest of the work easily gives a work factor of 2^{36} IDEA-X encryptions.

Many other variants of IDEA are also vulnerable to multiplicative differential attacks. Table 3 characterizes a large class of weak IDEA variants by showing the minimum changes necessary to IDEA to render it vulnerable. These IDEA variants have three different round functions, A, B, and C. The output function, D, is exactly as in the original IDEA cipher. The cipher can be any number of rounds, and can begin with A, B, or C, but must cycle through the rounds in the order A, B, C. The round functions are almost identical to the IDEA round functions, except that some of the additions have been changed to xors or multiplications. Each of the speci£ed additions may be replaced with either an xor or a multiply, independent of the other additions. The other additions in the cipher may also be replaced, but this isn't necessary.

This class of weak IDEA variants generalizes our results on IDEA-X: it is only necessary to remove half the additions from the cipher to render it vulnerable to multiplicative differential attacks. From this we conclude that multiplicative differentials can be applicable even to some ciphers with three incompatible group operations.

Cipher	Rounds	Probability	Pairs	Right pairs [Expected]	[Actual]
Nimbus	5	2^{-5}	10^6	31250	31245
xmx (standard version)	8	1	10^5	10^5	10^5
xmx (challenge version)	4	2^{-16}	10^6	15.3	562
IDEA-X	4	2^{-16}	10^8	1525.9	1537
MultiSwap	all	2^{-8}	10^8	390625	390532

Table 4. Experimental veri£cation of differential probabilities. We use reduced-rounds variants of the xmx challenge cipher and IDEA-X to make the measurement feasible.

8 Experimental Veri£cation

We performed several experiments to verify our claims. We £rst tested the differential probabilities derived in this paper; Table 4 summarizes the results. With the exception of the xmx challenge cipher, all the measurements agree with the theory.

The experiments show that the differential -1 survives 4 rounds of xmx with much higher probability than expected. Part of this discrepancy can be explained by observing that a randomly chosen weak key may allow many $-1 \leftrightarrow \Delta$ correspondences, for different choices of Δ, increasing the probability of the -1 differential. We did further experiments to verify that this was indeed the source of the discrepancy, and were able to predict the experimentally observed probability for a given key to within a factor of 4 (for 4 rounds). We leave it as an open question to explain the remaining error. Since the differential actually survives four rounds with probability about 2^{-11}, we estimate that the xmx challenge cipher can be distinguished using only 2^{23} chosen plaintexts.

We next veri£ed the claim, made in Section 5, that approximately 2^{-8} keys for the xmx challenge cipher are weak. Recall that a key s is weak if $s \wedge \neg\Delta^n = s^{-1} \wedge \neg\Delta^n = 0$. These comprise 4 bit constraints on s and 4 bit constraints on s^{-1}. It is not clear that this will be satis£ed by 2^{-8} keys, so we tested 10^6 randomly generated keys, from which we expected to £nd 3906 weak keys. The actual number of weak keys was 3886, con£rming our analysis.

Next we implemented the chosen-plaintext key recovery attack on IDEA-X. In order to make the data requirements feasible, we only attacked 4 rounds of IDEA-X, and only ran 10 trials. The IDEA-X experiments required an average of 124 right pairs to recover the key, about 4 times as many right pairs as we predicted in Section 7. Observing the attack in action reveals that frequently a small number of right pairs—around 30 or 40—are suf£cient to eliminate all but 2 or 3 candidates for a particular subkey. A more ef£cient attack would simply try each candidate, an approach we did not implement.

We implemented the known-plaintext attack on MultiSwap, but since this attack involves repeating the same attack on the two halves of the cipher, we only attacked the latter half. The attack worked as described in Section 6; however, on some trials there were too few right pairs to perform the analysis. Nonetheless, 70 out of 100 runs using 2^{22} known plaintexts were able to successfully recover the key. Increasing the number of plaintexts to 5000000 increased the success rate to 99%. Recall that the attack required guessing the correct value of k_{11} from a list sorted by likelihood. The average position of the correct k_{11} in this list was 2.

9 Conclusion

In this paper we have de£ned the concept of a multiplicative differential. We described several particular differentials and analyzed how they interact with standard operations used in cryptography such as xor and bit permutations. We then used these differentials to cryptanalyze two existing ciphers and variants of IDEA.

Our results demonstrate that the modular multiplication operation by itself is insuf£cient to prevent differential attacks. Further, multiplicative differentials can be surprisingly resilient in the presence of incompatible group operations. Therefore, multiplication needs to be carefully combined with other group operations to destroy these differential properties. We are hopeful that this paper will help further the understanding of how to use the multiply operator to build secure cryptographic algorithms.

Acknowledgements

We thank Joan Daemen for providing LaTeX source for a diagram of IDEA from which Figure 4 is derived. We also thank John Kelsey for suggesting that we implement our attacks, an exercise which led to many improvements to our results.

References

1. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
2. Joan Daemen, Rene Govaerts, and Joos Vandewalle. Weak keys for IDEA. In *CRYPTO*, pages 224–231, 1993.
3. Joan Daemen, Luc van Linden, Rene Govaerts, and Joos Vandewalle. Propagation properties of multiplication modulo $2^n - 1$. In G. H. L. M. Heideman et.al., editor, *Thirteenth Symp. on Information Theory in the Benelux*, pages 111–118, Enschede (NL), 1-2 1992. Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL).
4. Vladimir Furman. Differential cryptanalysis of Nimbus. In *Fast Software Encryption*. Springer-Verlag, 2001.
5. Carlo Harpes, Gerhard G. Kramer, and James L. Massey. A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-up Lemma. In *EUROCRYPT '95*. Springer-Verlag, May 1995.
6. John Kelsey, Bruce Schneier, and David Wagner. Mod n cryptanalysis, with applications against RC5P and M6. In *Fast Software Encryption*, pages 139–155, 1999.
7. Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In *EUROCRYPT '91*. Springer-Verlag, 1991.
8. Alexis Warner Machado. The Nimbus cipher: A proposal for NESSIE. NESSIE Proposal, September 2000.
9. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *EUROCRYPT '93*, volume 765, pages 386–397, Berlin, 1994. Springer-Verlag.
10. Willi Meier. On the security of the IDEA block cipher. In *EUROCRYPT '93*, pages 371–385. Springer-Verlag, 1994.
11. David M'Raihi, David Naccache, Jacques Stern, and Serge Vaudenay. XMX: a £rmware-oriented block cipher based on modular multiplications. In *Fast Software Encryption*. Springer-Verlag, 1997.
12. Beale Screamer. Microsoft's digital rights management scheme—technical details. `http://cryptome.org/ms-drm.htm`, October 2001.